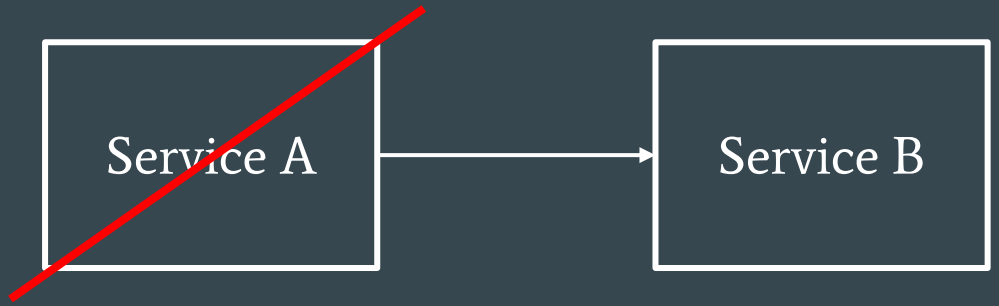
Migrations done right: Success stories and lessons learned

...

Fernando Cristiani - LeadDev Berlin 2025

Large Scale Migrations ...

- Leading a migration project will be part of your career at some point
  - A programming language isn't supported anymore in your company
  - A technology is no longer part of your company's tech radar
  - A vendor is no longer desired
  - A design or architecture is no longer a good fit
  - ...
- Running migrations successfully allows companies to embrace modernization projects



... are Difficult

- Run over a long period of time
- Very hard to estimate
- Scope is unpredictable
- Keeping yourself and your team motivated throughout the journey is challenging
- Explicit expectations management and building trust is crucial

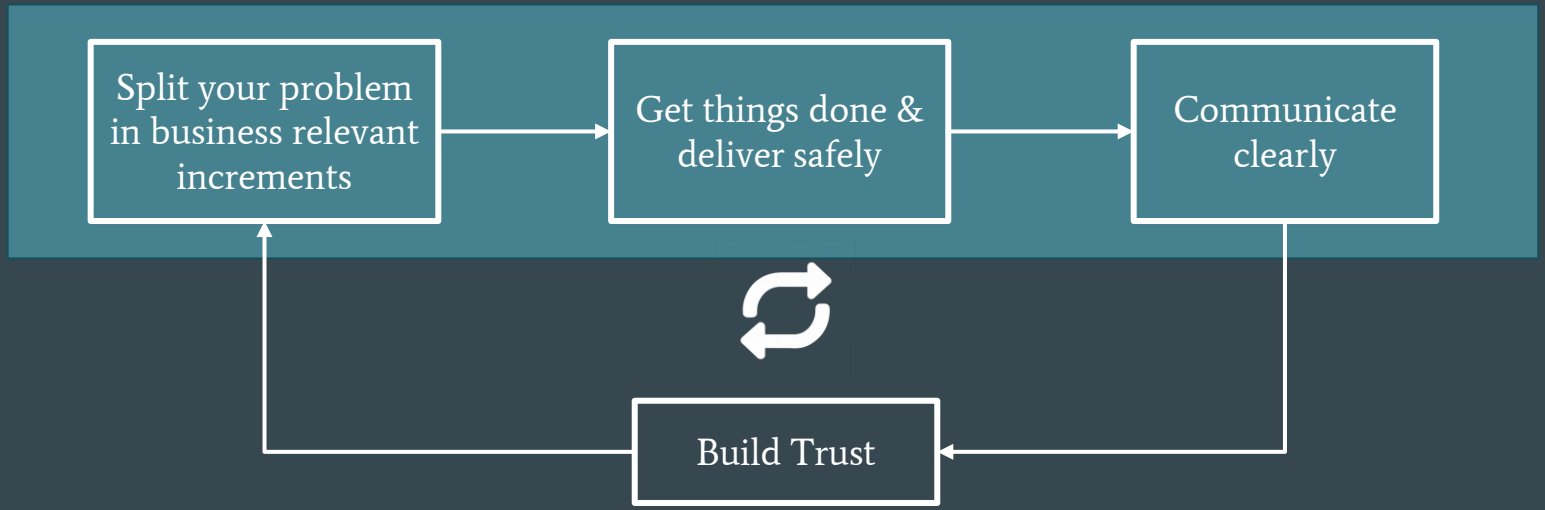
All Eyes on You

- Your migration project is over schedule and/or budget
- Your team starts becoming the bottleneck
- You are in the spotlight
- a.k.a. “The Eye of Sauron” [1]

[1] James Stanier - “Become an Effective Software Engineering Manager” & “Become a Great Engineering Leader”

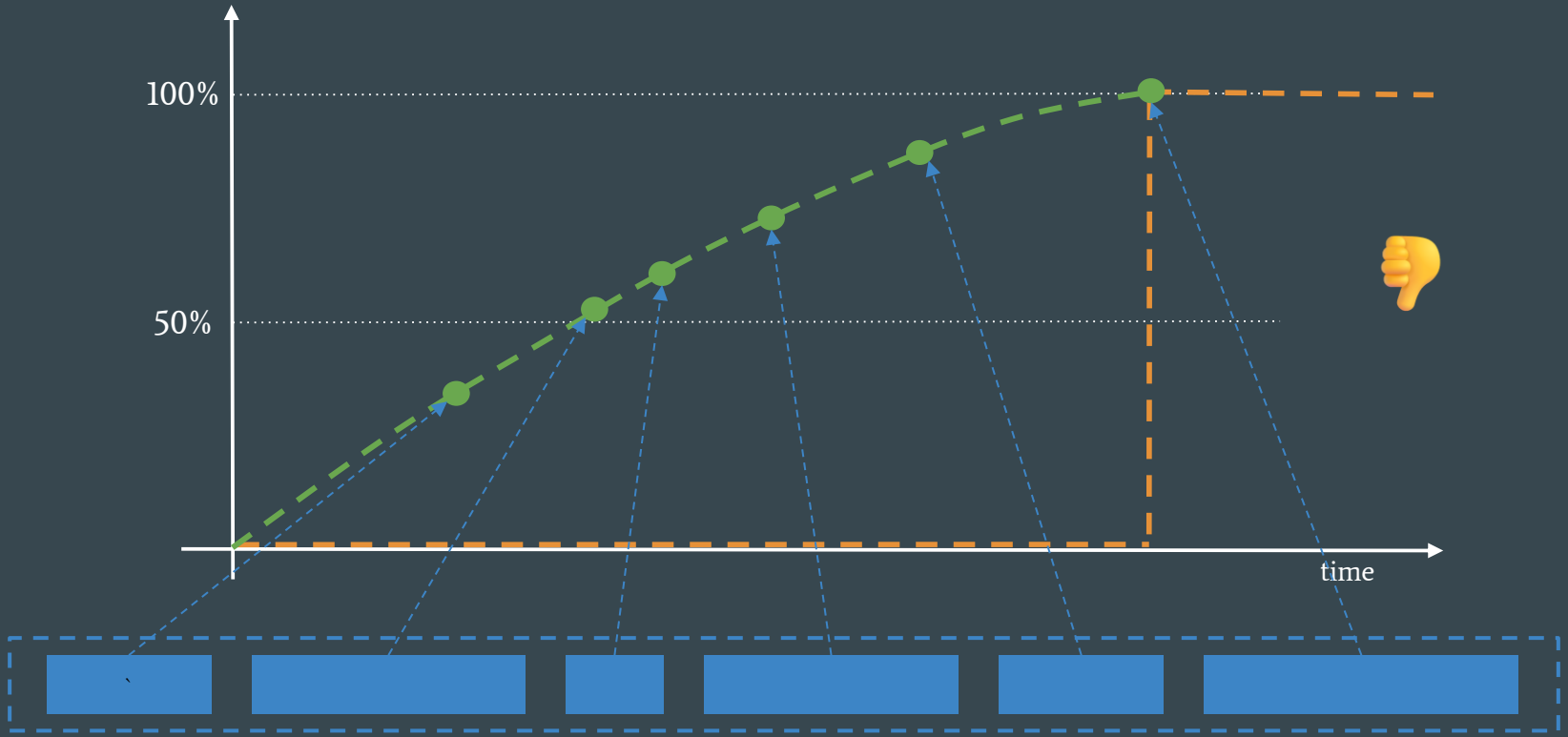
It’s All About Trust

- “Getting things done consistently over a long period of time” as the main source for building trust



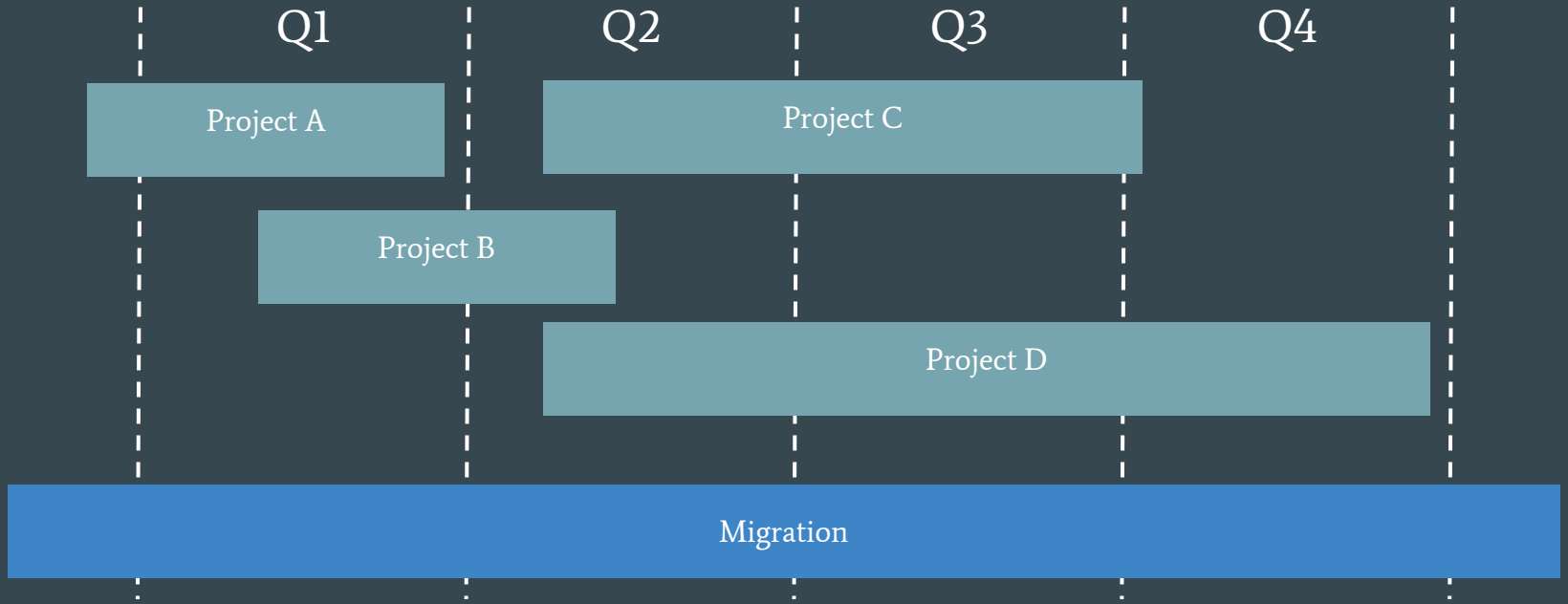
Splitting your migration

Splitting your migration



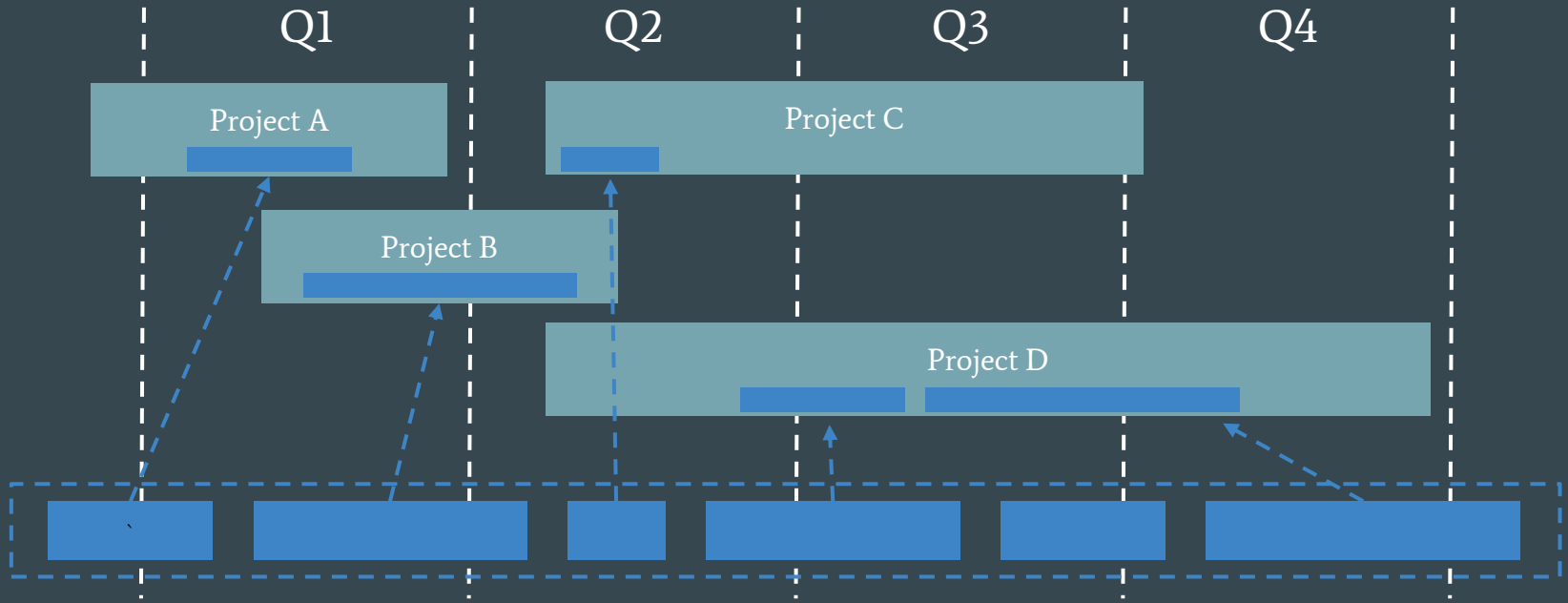
Splitting your migration

- How do you place the migration in your company's roadmap?
- Placing it as a standalone component can make it harder for you to build trust



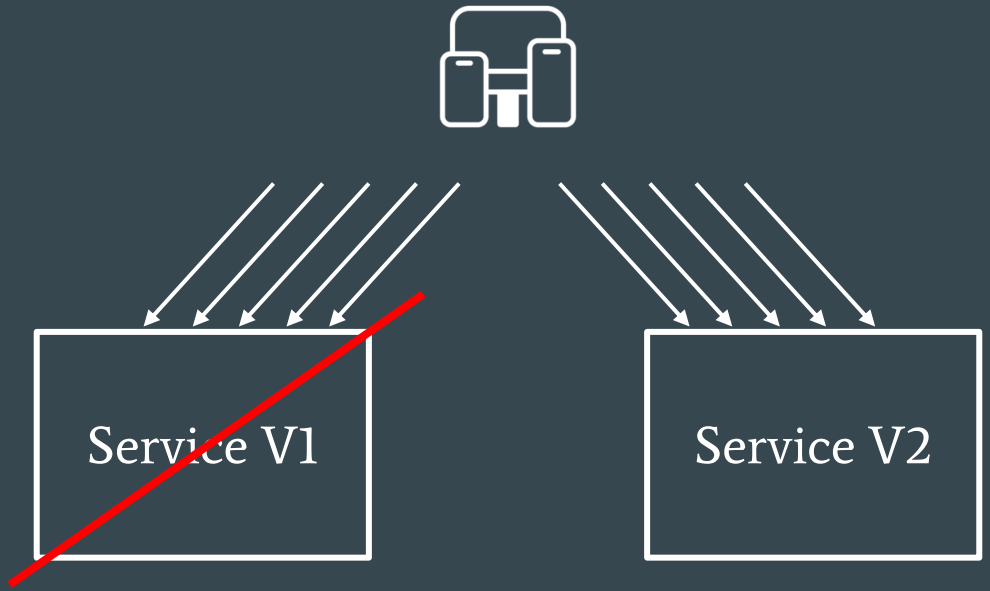
Splitting your migration

- Break your problem in a way that aligns with the roadmap
- Leverage high-priority projects to advance your migration



Splitting your migration

- Accept that you will have 2 systems running in parallel



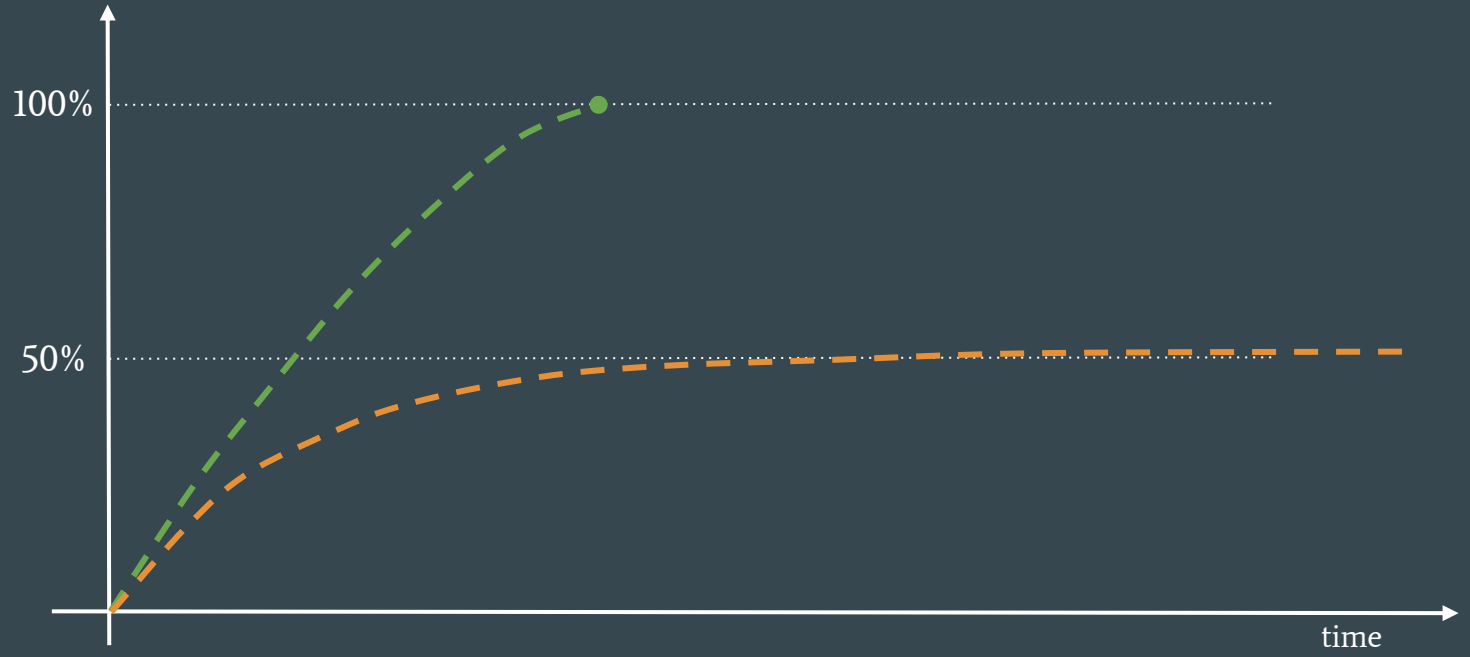
Splitting your migration

- Begin With the End in Mind [2]
  - Define your final goal
  - Make sure they contribute towards the goal in a genuine way

[2] Stephen R. Covey- “The 7 Habits of Highly Effective People”

Splitting your migration - Rain-drop migrations

- Organic distributed on-the-fly migrations triggered by existing use cases
- Make sure you know when you are finished



Splitting your migration - Rain-drop migrations

- Good fit for use cases with expiration or regular cycles
  - Payments or subscriptions
  - Session management
- A pragmatic and simple way of migrating the most important use cases first

Delivering Safely

Delivering Safely

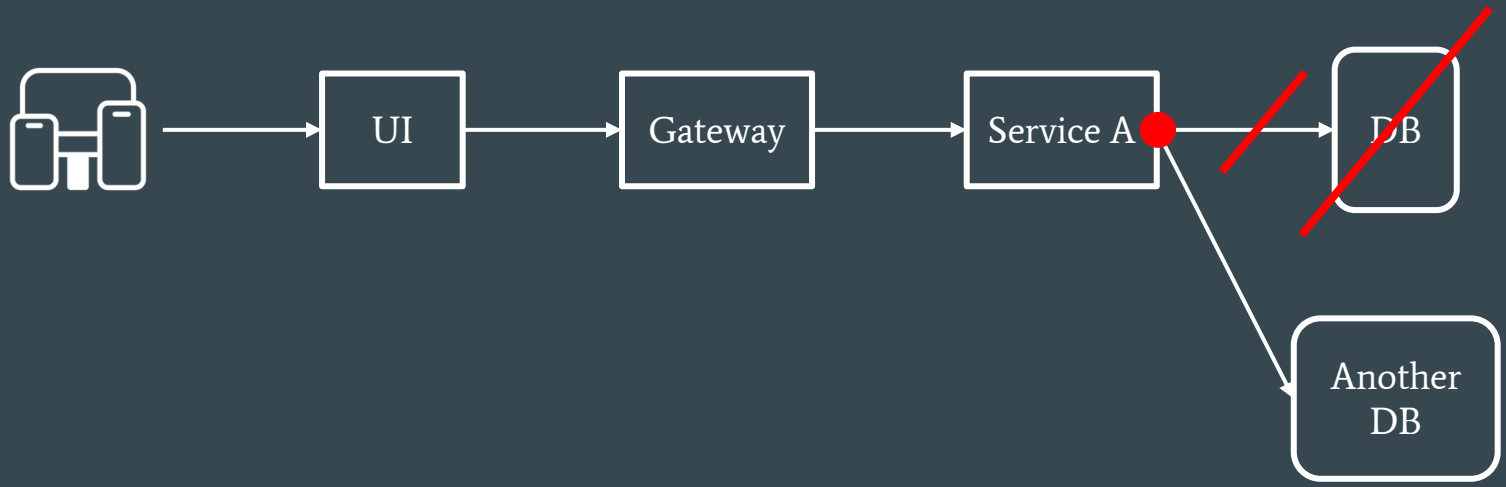
- Release on non-prod environments
- Release the code to prod with feature flag off
- Enable it in prod for some whitelisted users

- Gradually roll it out to more users
- Monitor and verify

- Once it is release to all users, remove the feature flag

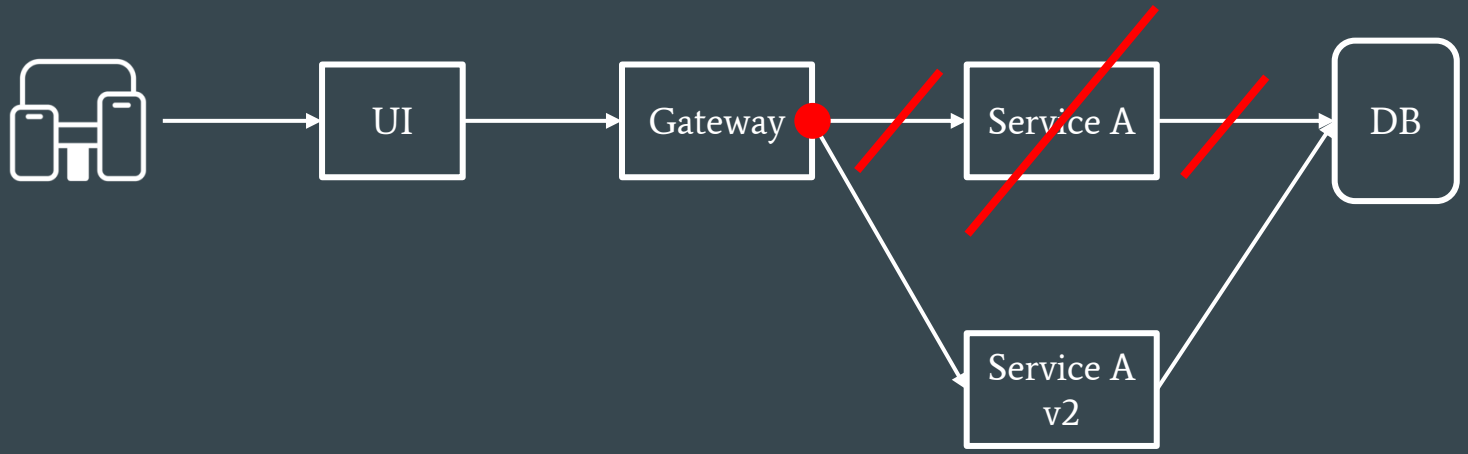
Delivering Safely - Splitting traffic

- Feature flags & repository pattern



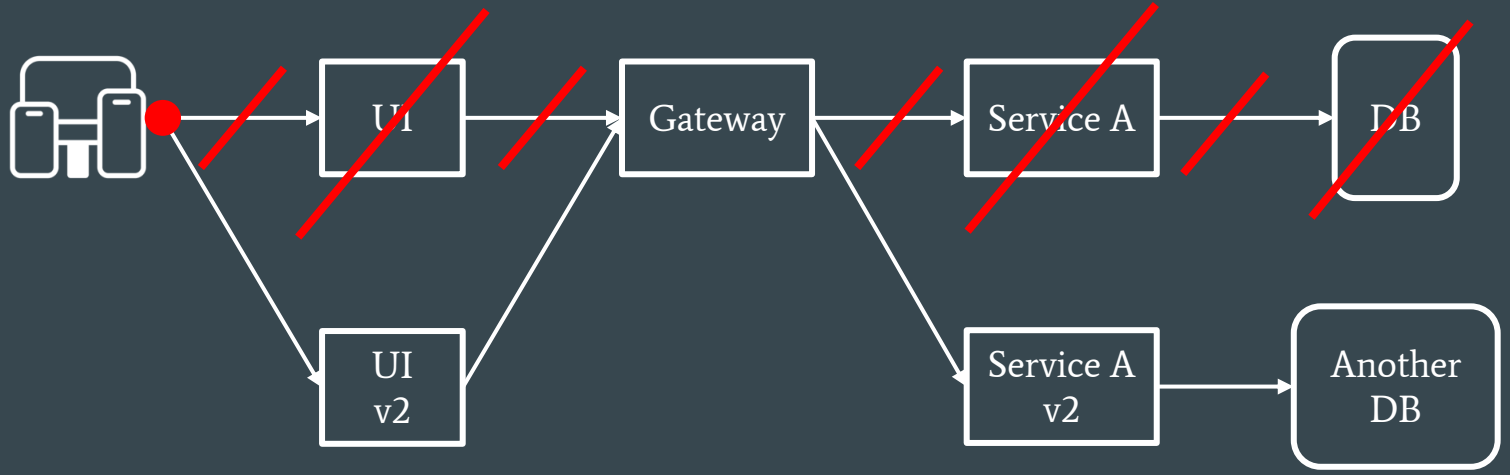
Delivering Safely - Splitting traffic

- AWS Cloudfront + Lambda@Edge
- Feature flags at Gateway level



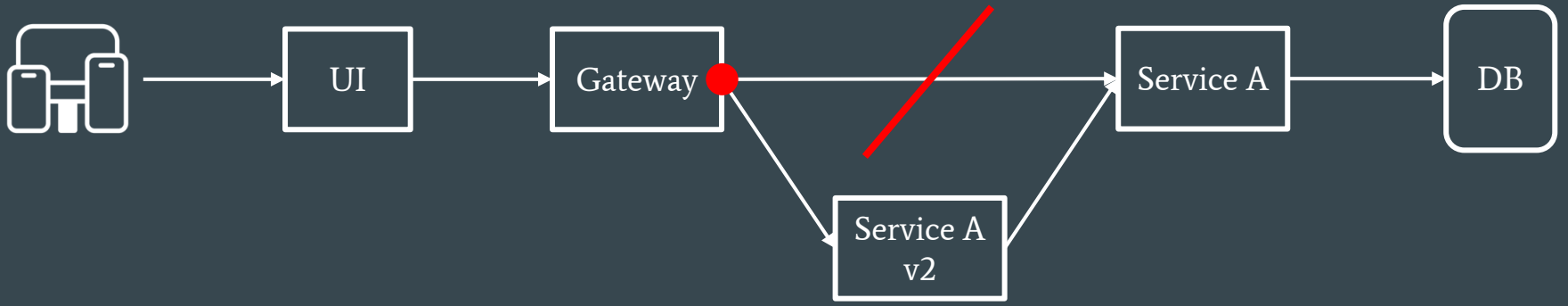
Delivering Safely - Splitting traffic

- Backend feature flag
- User routed to a different UI



Delivering Safely - Splitting traffic

- Passthrough



Showing progress

Showing progress

- Visibility is essential. Anyone should be able to understand the progress of the migration at any time
- Proactively communicate progress on a regular basis
- You might have to build custom tools to show progress
- Make sure to choose a variable that shows progress in an accurate way

Wrapping Up

Don’t forget about these

- Backups
- Decommission any unused service
- Delete unused code
- Delete all feature flags and any code associated with it
- Delete any leftovers of your migration framework
- Communicate explicitly and broadly the end of the migration
- **Celebrate the success**

Lessons Learned and Key Takeaways

- Sustainable & pragmatic software as the main tool to avoid migrations
  - Even if you do migrate them: Good codebases are easier to migrate

- Very long running projects - Building trust and keeping it is essential
  - Split your problem, get things done & deliver, and communicate effectively
  - Decide when the migration is finished. Take control of the narrative.

- You will have 2 systems running in parallel for a while:
  - Don’t underestimate the cost and effort for this: make sure it’s not indefinitely
  - Treat *both* of them as production systems
  - Treat your migration framework as production code

Lessons Learned and Key Takeaways

- Lead with Clarity
  - Set a clear final goal and a way forward. Make it visible and repeat it.
- You will need a strong team
  - Make sure you have the right team setup
  - Keep your team motivated: make progress visible and celebrate
- Make sure to have the right feedback loops and support
  - Detect when you are in the rabbit hole
  - Get help to get out of it
- Always invest in your reputation “savings account”. It can save you when things go sideways

Thank you